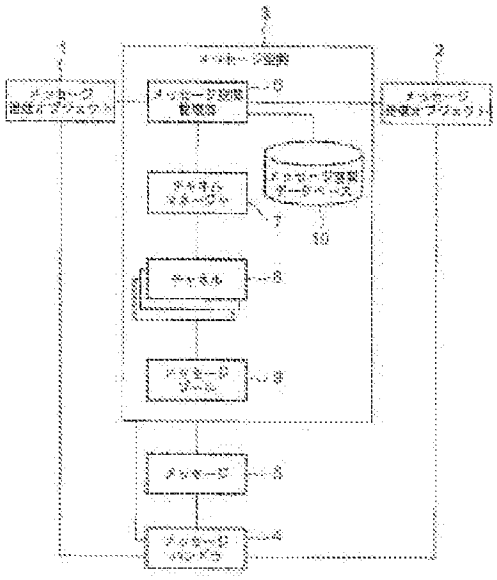
MASSAGING PROGRAM, MESSAGING METHOD OF DISTRIBUTED SYSTEM, AND MESSAGING SYSTEM

Publication number: JP2002269063 (A)
Publication date: 2002-09-20
Inventor(s): ANDO TAKANOBU; YANO REI; AKIMOTO NAOTO; KURATA TOMOMI +
Applicant(s): TOSHIBA CORP +
Classification:
- international: ***G06F13/00; G06F15/16; G06F15/177; G06F9/46; G06F9/54;*** **G06F13/00**; **G06F15/16**; **G06F9/46**; (IPC1-7): G06F13/00; G06F15/16; G06F15/177
- European:
Application number: JP20010063985 20010307
Priority number(s): JP20010063985 20010307

Abstract of **JP 2002269063 (A)**

PROBLEM TO BE SOLVED: To actualize a mechanism for messaging which is usable as a reference model for a distributed system and has high flexibility in specification alteration. SOLUTION: This messaging program enables a computer to function as a message space managing means 6 which actualizes the management of at least one channel 8 receiving a message 5 from a message transmission side 1 and exchanging the message with a system corresponding to the nature of the message 5 and passes the received message 5 to the channel 8 corresponding to the nature of the message 5 so as to exchange messages between the message transmission side 1 and a message reception side 2 of the distributed system.



(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号
特開2002－269063
（P2002－269063A）
(43)公開日　平成14年９月20日(2002.9.20)

(51)Int.Cl.[7]		識別記号	FI		テーマコード（参考）
Ｇ０６Ｆ	15/177	６７６	Ｇ０６Ｆ 15/177	６７６Ｚ	５Ｂ０４５
				６７６Ｃ	５Ｂ０８９
	13/00	３５３	13/00	３５３Ａ	
	15/16	６２０	15/16	６２０Ｔ	

審査請求　未請求　請求項の数10　ＯＬ　（全 26 頁）

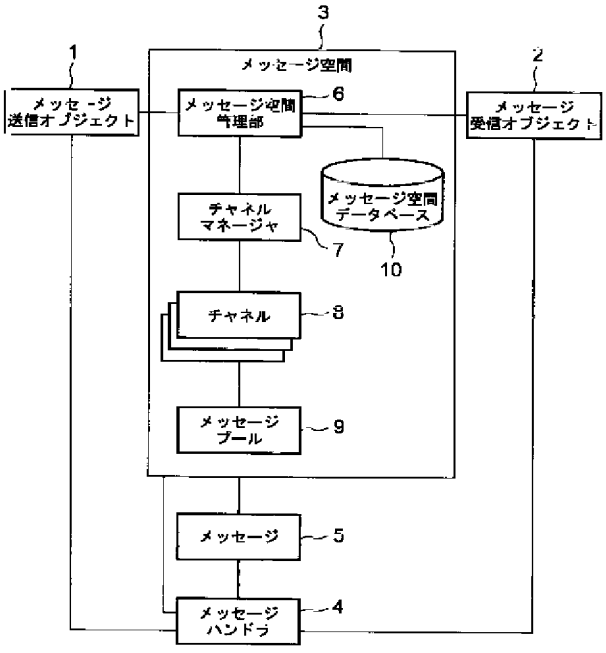
(21)出願番号　特願2001－63985(P2001－63985)
(22)出願日　平成13年３月７日(2001.3.7)

(71)出願人　000003078
株式会社東芝
東京都港区芝浦一丁目１番１号
(72)発明者　安東　孝信
東京都府中市東芝町１番地　株式会社東芝府中事業所内
(72)発明者　矢野　令
東京都府中市東芝町１番地　株式会社東芝府中事業所内
(74)代理人　100058479
弁理士　鈴江　武彦　（外６名）

最終頁に続く

(54)【発明の名称】　メッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステム

(57)【要約】
【課題】分散システム向けの参照モデルとして利用可能であり、さらに仕様変更に対して高い柔軟性を持つメッセージングの仕組みを実現する。
【解決手段】分散システムにおけるメッセージ送信側１とメッセージ受信側２との間のメッセージ交換のためにコンピュータを、メッセージ送信側１からのメッセージ５を受け付け、メッセージ５の種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネル８の管理を実現し、受け付けたメッセージ５をそのメッセージ５の種類に応じたチャネル８に渡すメッセージ空間管理手段６として機能させるためのメッセージングプログラム。



【特許請求の範囲】
【請求項１】　分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のために、コンピュータに、
前記メッセージ送信側からのメッセージを受け付けさせる受付手段と、
前記受付手段で受け付けた前記メッセージをそのメッセージの種類に応じたチャネルに渡させるメッセージ空間管理手段とを有し、
前記チャネルは、前記メッセージの種類に応じた方式で前記メッセージ交換を制御することを特徴とするメッセージングプログラム。
【請求項２】　分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためにコンピュータを、
前記メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡すメッセージ空間管理手段として機能させるためのメッセージングプログラム。
【請求項３】　請求項２記載のメッセージングプログラムにおいて、
前記メッセージ空間管理手段は、定義された共通のインタフェースとその利用手順にしたがってアクセスされることを特徴とするメッセージングプログラム。
【請求項４】　請求項２又は請求項３記載のメッセージングプログラムにおいて、
前記メッセージ空間管理手段は、前記メッセージ送信側を宛先とする送達確認メッセージを前記メッセージ受信側から受け付け、この受け付けた送信確認メッセージをその種類に応じたチャネルに渡すことを特徴とするメッセージングプログラム。
【請求項５】　請求項１乃至請求項４のいずれか１項記載のメッセージングプログラムにおいて、
前記メッセージ空間管理手段は、メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供することを特徴とするメッセージングプログラム。
【請求項６】　分散システムにおけるメッセージ送信側とメッセージ受信側との間でメッセージ交換を行うためのメッセージング方法において、
前記メッセージ送信側からメッセージを受け付ける工程と、
前記メッセージ送信側からのメッセージの種類に応じた方式でメッセージ交換を制御するチャネルがない場合に、前記メッセージ送信側からのメッセージの種類に対応するチャネルを生成する工程と、
前記メッセージ送信側からのメッセージをその種類に対応するチャネルに渡す工程と、
前記メッセージ送信側からのメッセージを受けたチャネルにより、前記メッセージ送信側からのメッセージを前記メッセージ受信側に配信する工程とを含むことを特徴とする分散システムにおけるメッセージング方法。
【請求項７】　請求項６記載の分散システムにおけるメッセージング方法において、
定義された共通のインタフェースとその利用手順にしたがってメッセージが交換されることを特徴とする分散システムにおけるメッセージング方法。
【請求項８】　請求項６又は請求項７記載の分散システムにおけるメッセージング方法において、
メッセージを受信した前記メッセージ受信側から送達確認メッセージを受け付ける工程と、
前記送達確認メッセージの種類に応じた方式でメッセージ交換を制御するチャネルがない場合に、前記送達確認メッセージの種類に対応するチャネルを生成する工程と、
前記送達確認メッセージをその種類に対応するチャネルに渡す工程と、
前記送達確認メッセージを受けたチャネルにより、前記送達確認メッセージを前記メッセージ送信側に配信する工程とを含むことを特徴とする分散システムにおけるメッセージング方法。
【請求項９】　請求項６乃至請求項８のいずれか１項記載の分散システムにおけるメッセージング方法において、
メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供することを特徴とする分散システムにおけるメッセージング方法。
【請求項１０】　分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためメッセージングシステムにおいて、
前記メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡すメッセージ空間管理手段を具備したことを特徴とするメッセージングシステム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、分散システムにおいてメッセージを交換するためのメッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステムに関する。
【０００２】
【従来の技術】分散システムでは、複数の計算機に処理を分散させる。オブジェクト指向技術を利用して分散システムを構築した場合、指示命令やデータなどの情報であるメッセージが分散されているオブジェクト間でやり取りされる。

【０００３】分散システムのメッセージングシステムとして、CORBAのイベントサービスや、JavaRMI（Java Remote Method Invocation）、ECJ（Event Centric for Java）などいくつかの代表的な基盤技術が存在する。

【０００４】

【発明が解決しようとする課題】CORBAとは、分散システムでオブジェクト同士がメッセージを交換するためのソフトウェアである。

【０００５】CORBAのイベントサービスでは、サプライヤとコンシューマとイベントチャネルからなるイベント通信モデルが提供されている。またイベント通信のスタイルとしてpush型とpull型、そしてこれらの混合型の３種類が定められている。

【０００６】このように、CORBAでは基本的なメッセージングの機能や枠組みは提供されている。しかしながら、アプリケーションレベルでの送達確認機能や、イベントチャネルに対するイベントの検索機能、例えばイベントの消失などのようなエラーに関する処理などについては明確に定義されていない。また、CORBAは様々なメーカーから提供されているが、イベントサービスを利用したアプリケーションの実装方法はそれぞれによって異なる。

【０００７】JavaRMIは、リモートにあるJavaオブジェクトのメソッドを他のホストのJavaVM（Java Virtual Machine）から呼び出すための分散オブジェクト技術である。これにより、異なるJavaVM上で実行中のオブジェクト同士の間でメッセージ交換が可能になる。Java RMIは基本的に同期通信に分類されるため、非同期通信の実装に直接用いることはできない。

【０００８】Javaベースの非同期通信では、JMS(Java Messaging Service)がある。JMSではPeer-to-peerとPublish/Subscribeの２通りの通信モデルが提供されている。

【０００９】しかしながら、JMSでも、CORBAと同様にアプリケーションレベルでの送達確認機能や通信路であるConnection/Sessionに対するメッセージの検索機能、メッセージ消失などのエラーに対する処理などは提供されない。

【００１０】ECJもJavaベースのイベント通信技術である。ECJは、インターネット／イントラネット上に分散したイベント駆動型のJavaアプリケーションを構築するためのフレームワークであり、RMIを補完する非同期メッセージ交換機構を提供する。ECJではSocketを利用することで非常に軽量で高速なイベント通信を可能としている。

【００１１】しかしながら、ECJでも上記と同様にアプリケーションレベルでの送達確認機能、通信路におけるメッセージの検索機能、メッセージ消失などのエラーに対する処理などは提供されていないという課題を持つ。

【００１２】また上述したそれぞれのメッセージング技術は、それぞれで実装方法が異なる。そのため、一般に各アプリケーションの実装は利用する基盤技術に依存してしまい、基盤技術の交換が大変困難である。また、基盤技術毎にメッセージングのアーキテクチャが異なるため、ある基盤技術の実装で培った経験を、他の基盤技術に対して活用しにくいという問題も生じている。

【００１３】特に、従来においては、分散システム向けメッセージングのための参照モデル／フレームワークが存在しない。ここで、フレームワークとは、システムやアプリケーションを組み上げる場合に参照する大枠の構成要素の概念、あるいはこの概念を具体化したソフト基盤をいう。フレームワークを利用すればアプリケーション間の連携を標準化、統合化できるが、従来のメッセージングの技術ではこのフレームワークが存在しないため標準化、統合化が困難である。

【００１４】本発明は、以上のような実情に鑑みてなされたもので、分散システム向けメッセージングのための参照モデルとして利用可能であり、さらに仕様変更に対して高い柔軟性を実現できるメッセージングプログラム、及び分散システムにおけるメッセージング方法、並びにメッセージングシステムを提供することを目的とする。

【００１５】

【課題を解決するための手段】以下、本発明を実現するにあたって講じた具体的手段について説明する。

【００１６】本発明は、分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のために、コンピュータに、メッセージ送信側からのメッセージを受け付けさせる受付手段と、受付手段で受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡させるメッセージ空間管理手段とを有し、チャネルは、メッセージの種類に応じた方式でメッセージ交換を制御することを特徴とするメッセージングプログラムである。

【００１７】また、本発明は、分散システムにおけるメッセージ送信側とメッセージ受信側との間のメッセージ交換のためにコンピュータを、メッセージ送信側からのメッセージを受け付け、メッセージの種類に応じた方式でメッセージ交換を制御する少なくとも一つのチャネルの管理を実現し、受け付けたメッセージをそのメッセージの種類に応じたチャネルに渡すメッセージ空間管理手段として機能させるためのメッセージングプログラムである。

【００１８】本発明を利用し、基盤技術毎に異なるメッセージング実装のための手段や仕組みを統一化することで、適切な参照モデル／フレームワークを提供でき、分散システム向けメッセージングのための構成を明確化できる。また、ある基盤技術の実装で培った経験を他の基盤技術で活用することが容易になる。

【００１９】また、本発明で採用している参照モデル／

フレームワークは、例えばCORBA、JavaRMI、JavaとECJなどのような既存の分散システム間通信のための基盤技術を用いる場合にも実装可能である。

【００２０】なお、チャネルは、チャネルマネージャによって管理され、このチャネルマネージャがさらにメッセージ空間管理手段によって管理されるとしてもよい。

【００２１】本発明のメッセージ空間管理手段は、定義された共通のインタフェースとその利用手順（プロトコル）にしたがってアクセスされるように変形可能である。このような変形により、分散システムにおけるメッセージングのためのインタフェースやプロトコルを共通化できる。

【００２２】また、本発明のメッセージ空間管理手段は、メッセージ送信側を宛先とする送達確認メッセージをメッセージ受信側から受け付け、この受け付けた送信確認メッセージをその種類に応じたチャネルに渡すように変形可能である。このような変形により、メッセージの送達確認機能を付加できる。

【００２３】さらに、本発明のメッセージ空間管理手段は、メッセージ交換の履歴情報を記録し、外部からの問い合わせに応じて該当する履歴情報を提供するように変形可能である。このような変形により、履歴情報に基づく処理を実行してメッセージ空間に対するメッセージの検索機能、メッセージ不着信検出及びその対処機能を提供できる。

【００２４】上記のようなメッセージングプログラム又はメッセージングプログラムを記録した記録媒体を用いることによって、分散システムにおけるメッセージ交換に対して簡単に上記の作用効果を実現できる。

【００２５】また、本発明を利用すれば、メッセージ送信側からメッセージを受け付ける工程と、メッセージ送信側からのメッセージの種類に応じた方式でメッセージ交換を制御するチャネルがない場合に、メッセージ送信側からのメッセージの種類に対応するチャネルを生成する工程と、メッセージ送信側からのメッセージをその種類に対応するチャネルに渡す工程と、メッセージ送信側からのメッセージを受けたチャネルにより、メッセージ送信側からのメッセージをメッセージ受信側に配信する工程とを含む分散システムにおけるメッセージング方法を実施可能である。

【００２６】この分散システムにおけるメッセージング方法についても上述した変形と同様の変形を行ってもよい。

【００２７】また、本発明の各手段を具備したメッセージ交換システムを発明の対象としてもよい。

【００２８】

【発明の実施の形態】以下、図面を参照しながら本発明の実施の形態について説明する。

【００２９】（第１の実施の形態）本実施の形態においては、分散システムのためのメッセージングシステムについて説明する。このメッセージングシステムは、メッセージングの仕組み、インタフェース定義、システム利用者からみた利用シーケンス、参照モデル／フレームワーク、エラーハンドリングに特徴がある。

【００３０】図１は、本実施の形態に係るメッセージングシステムを利用する分散システムの構成を例示するブロック図である。

【００３１】この分散システムは、メッセージ送信オブジェクト（アプリケーション）１、メッセージ受信オブジェクト（アプリケーション）２、メッセージ空間３、メッセージハンドラ４から構成されている。メッセージ送信オブジェクト１とメッセージ受信オブジェクト２との間でメッセージ５が交換される。

【００３２】メッセージ送信オブジェクト１は、メッセージ５を作成し、メッセージ空間３に作成したメッセージ５を送信する。オプションで送達確認用のメッセージハンドラ４を用意し、メッセージ空間３に登録可能である。メッセージ受信オブジェクト２はメッセージ空間３からメッセージを受信する。受信方法には、受信用のメッセージハンドラ４をメッセージ空間３に登録し、それを用いてメッセージ空間３から呼び出されるようにする形式のpush型と、メッセージ受信オブジェクト２が自発的にメッセージ空間３にポーリングをかける形式のpull型の２通りの方式がある。なお、以下においては、どちらのメッセージの受け取り形式が用いられたとしても「配送する」と表記する。

【００３３】メッセージ空間３は、メッセージ送信オブジェクト１から受け付けたメッセージ５をメッセージ受信オブジェクト２へ配送するための概念的な存在である。

【００３４】メッセージ５はメッセージ送信オブジェクト１がメッセージ受信オブジェクト２に配送するために必要とする情報を含むデータである。

【００３５】メッセージハンドラ４は、対応するメッセージ５がメッセージ空間３に受け付けられた場合に、そのメッセージ交換を要求するメッセージ送信オブジェクト１やメッセージ受信オブジェクト２をコールバックし、メッセージを受け渡すために使われるインタフェースを実装した構成要素である。

【００３６】また、メッセージハンドラ４は、予定されているメッセージ５の不着が検出された際のインタフェースも備える。

【００３７】メッセージ空間３はさらにメッセージ空間管理部６とチャネルマネージャ７、チャネル８、メッセージプール９、そしてメッセージ空間データベース１０から構成される。

【００３８】メッセージ空間管理部６は、メッセージ空間３全体に関わる処理、例えばメッセージ空間３の初期化や、メッセージ５の登録受付、チャネル８やチャネルマネージャ７の準備、メッセージハンドラ４の受付、送

信されたメッセージ５のチャネル８への割り当てなどを行う。

【００３９】メッセージ空間管理部６は、これらの処理を実現するために、メッセージ空間３内でやり取りされるメッセージ５の種類、メッセージの種類とチャネル８の対応関係、メッセージの種類とチャネルマネージャ７の対応関係、メッセージやメッセージの種類とメッセージ送信オブジェクト１との対応関係、メッセージやメッセージの種類とメッセージ受信オブジェクト２との対応関係などに関する情報を、メッセージ空間データベース１０を用いて保持している。

【００４０】メッセージ送信オブジェクト１やメッセージ受信オブジェクト２がメッセージ空間３へアクセスするためのインタフェースを提供するのもメッセージ空間管理部６である。

【００４１】チャネルマネージャ７は、チャネル８に対する初期化や複数のチャネル８間の制御を行う。

【００４２】チャネル８は、メッセージ５の種類毎に用意され、メッセージ空間３内でのメッセージ５の交換（やり取り）を制御する。

【００４３】メッセージプール９は、メッセージ５の送信先（受け手）に一対一で対応する。このメッセージプール９は、メッセージ５を蓄積しておくための構成要素である。なお、メッセージプール９はオプションである。

【００４４】チャネルマネージャ７は、図２に示すように、例えばシナリオチャネルマネージャ１１などのようなチャネル８に対して特別な制御を行う構成要素を派生する。シナリオチャネルマネージャ１１は、例えば、指定された複数種類のメッセージ５の順序を矛盾なく配送するための制御を行う。

【００４５】チャネル８は、図３に示すように、キューチャネル１２やストアチャネル１３、シンプルチャネル１４などのようなメッセージ５に対する扱いの種類に応じた構成要素を派生する。

【００４６】例えば、キューチャネル１２は、図４に示すように、メッセージ送信オブジェクト１からメッセージ５を受け取ると、メッセージ５を一旦キュー１５へ格納してからメッセージ受信オブジェクト２へ配送する。

【００４７】ストアチャネル１３は、図５に示すように、メッセージ送信オブジェクト１からメッセージ５を受け取ると、そのメッセージ５をメッセージ受信オブジェクト２に配送するとともに、ストア１６にも格納する。メッセージ受信オブジェクト２のメッセージ検索により、ストア１６内のメッセージ５はメッセージ受信オブジェクト２に配信される。

【００４８】シンプルチャネル１４は、図６に示すように、特にメッセージプール９は用いず、メッセージ空間３が受け付けたメッセージ５をメッセージ受信オブジェクト２にそのまま配送する。

【００４９】また、どのチャネル８もオプションで、送達確認をサポートするための機能（メッセージの送信先リスト、送信履歴、送達確認の受信履歴、これらの対応関係の監視など）を有している。

【００５０】ストアチャネル１３やシンプルチャネル１４はメッセージ５の配送のために、必要に応じて配送先のメッセージハンドラ４（のリスト）を保持する。また、必要に応じて送受信したメッセージ５の履歴やその送達確認情報の履歴についても保持する。

【００５１】メッセージプール９は、図７に示すように、キュー１５やストア１６などのようなメッセージ５を保持する方法の違いに応じた構成要素を派生する。

【００５２】例えば、キュー１５は、メッセージ５を受け取った順に、メッセージ受信オブジェクト２に配送する。これを実現するために、キュー１５は必要に応じて配送先のメッセージハンドラ４を保持する。

【００５３】ストア１６は、メッセージ５をランダムアクセス可能な記憶装置へ格納する。

【００５４】キュー１５はさらに、優先順位付きキュー１７、順序保証付きキュー１８、永続キュー１９などのサブ構成要素を派生する。

【００５５】例えば、優先順位付きキュー１７は、キュー１５に貯えられているメッセージ５のうちもっとも優先順位の高いものを優先してメッセージ受信オブジェクト２へ配送する。順序保証キュー１８は、指定された順序を優先してメッセージ５をメッセージ受信オブジェクト２へ配送する。永続キュー１９は配送したメッセージ５の履歴情報を残し、後からも検索処理などを実行可能にする。

【００５６】以下に、インタフェースの定義にについて説明する。

【００５７】メッセージ空間３は、メッセージ送信オブジェクト１やメッセージ受信オブジェクト２に公開するインタフェースに定義されたメソッド（操作）を通して、様々な機能を提供する。

【００５８】本実施の形態では、利用する通信基盤技術に依存しないインタフェースを定義する。

【００５９】メッセージ空間３が提供するメソッドはその目的に応じて利用開始、メッセージの送受信、検索処理、利用終了、管理の５種に分類される。

【００６０】以下に本実施の形態における定義をそれぞれ種類毎に列挙する。またメッセージハンドラが定義すべきメソッドについても定義する。

【００６１】利用開始のためのメソッド(A)：

(1)送信するメッセージ５のメッセージ空間３におけるコンテキストの取得

［定義］メッセージ空間３に対して、メッセージ送信オブジェクト１が送信しようとしている種類のメッセージ５のコンテキストを取得するためのメソッド、コンテキストは、指定されたメッセージ５が既にメッセージ空間

３内に存在していれば、そのメッセージ５を扱っているチャネルマネージャ７、チャネル８、メッセージプール９の種類を示す識別子群を含む。

【００６２】［シグネチャ］コンテキストを調べたいメッセージ５を引数としてもつ。返り値はコンテキストであり、メッセージ空間３内にまだ存在していなければ空の値を返す。

【００６３】［動作］このメソッドが呼ばれると、メッセージ空間３は、指定されたメッセージ５がメッセージ空間管理部６で既に管理してあるかどうかを、メッセージ空間データベース１０を用いて調べる。もし既に存在している場合には、そのメッセージ５を取り扱っているチャネルマネージャ７、チャネル８、メッセージプール９の種類を調べ、コンテキストとして返す。もし存在しない場合は空の値を返す。

【００６４】(2)送信するメッセージ５の種類を登録するメソッド（その１）

［定義］メッセージ送信オブジェクト１が送信するメッセージ５をメッセージ空間３に登録するためのメソッド。メッセージ５がまだメッセージ空間内で扱われていない場合に使うメソッド。メッセージ５の種類毎に登録する。

【００６５】［シグネチャ］登録するメッセージ５とチャネルマネージャ７、チャネル８の種類をそれぞれ指定するための識別子と、メッセージ送信オブジェクト１を特定するための識別子をそれぞれ引数としてもつ。

【００６６】［動作］このメソッドが呼ばれると、メッセージ空間３は、メッセージ５の種類とメッセージ送信オブジェクト１の識別子をメッセージ空間データベース１０へ記録するとともに、適切なチャネルマネージャ７を通して、与えられたメッセージ５のやり取りを担当するためのチャネル８を用意し、適切な初期化を行う。また、メッセージ送信オブジェクト１の識別子をチャネル８で記録する。さらに、チャネル８が送達確認用のメッセージ５のためのメッセージプール９としてキュー１５を用いている場合は、そのメッセージ送信オブジェクト１に対応するキュー１５を用意する。

【００６７】(3)送信するメッセージ５の種類を登録するメソッド（その２）

［定義］メッセージ送信オブジェクト１が送信するメッセージ５をメッセージ空間３に登録するためのメソッド、メッセージ５が既にメッセージ空間内で扱われている場合に使うメソッド、メッセージ５の種類毎に登録する。

【００６８】［シグネチャ］登録するメッセージ５とメッセージ送信オブジェクト１を特定するための識別子を引数としてもつ。

【００６９】［動作］このメソッドが呼ばれると、メッセージ空間３は、メッセージ５の種類とメッセージ送信オブジェクト１の識別子をメッセージ空間データベース１０へ記録するとともに、与えられたメッセージ５をやり取りしているチャネル８に対して、与えられたメッセージ送信オブジェクト１の識別子を記録する。さらに、チャネル８が送達確認用のメッセージ５のためのメッセージプール９としてキュー１５を用いている場合は、そのメッセージ送信オブジェクト１に対応するキュー１５を用意する。

【００７０】(4)送達確認をpush形式で受け取るための確認用ハンドラを登録するメソッド

［定義］メッセージ送信オブジェクト１が、送信するメッセージ５の送達確認を受けるためのメッセージハンドラ４をメッセージ空間３に登録するためのメソッド。メッセージ５の種類毎に登録する。

【００７１】［シグネチャ］このメソッドは登録するメッセージハンドラ４とそれが対応するメッセージ５を指定するための識別子を引数として持つ。

【００７２】［動作］このメソッドが呼ばれると、メッセージ空間３は、与えられたメッセージ５を担当するチャネル８に、確認用ハンドラとして与えられたメッセージハンドラ４を登録する。このチャネル８が送達確認用のメッセージ５のためのメッセージプール９としてキュー１５を用いている場合は、そのメッセージ送信オブジェクト１に対応するキュー１５に与えられたメッセージハンドラ４を登録する。

【００７３】(5)受信するメッセージ５の種類を登録するメソッド

［定義］メッセージ受信オブジェクト２が、配送を希望するメッセージ５をメッセージ空間３に登録するためのメソッド、メッセージ５の種類毎に登録する。

【００７４】［シグネチャ］このメソッドは登録するメッセージ５とメッセージ受信オブジェクト２を特定するための識別子を引数として持つ。

【００７５】［動作］このメソッドが呼ばれると、メッセージ空間３は、メッセージ５の種類とメッセージ受信オブジェクト２の識別子をメッセージ空間データベース１０へ記録するとともに、与えられたメッセージ５を担当するチャネル８に、メッセージ５の配送先として指定されたメッセージ受信オブジェクト２の識別子を伝える。チャネル８が送信用にキュー１５をメッセージプール９として用いている場合は、その配送先のための対応するキュー１５を作成する。

【００７６】(6)メッセージ５をpush形式で受け取るためのメッセージハンドラを登録するメソッド

［定義］メッセージ受信オブジェクト２が、配送を希望するメッセージ５を受け取るためのメッセージハンドラ４をメッセージ空間３に登録するためのメソッド。メッセージ５の種類毎に登録する。

【００７７】［シグネチャ］このメソッドは登録するメッセージハンドラ４とそれが対応するメッセージ５を指定するための識別子を引数として持つ。

【００７８】［動作］このメソッドが呼ばれると、メッセージ空間３は、メッセージ５の種類とメッセージ受信オブジェクト２の識別子をメッセージ空間データベース１０へ記録するとともに、与えられたメッセージ５を担当するチャネル８に、与えられたメッセージハンドラ４を登録する。チャネル８が送信用に、キュー１５をメッセージプール９として用いている場合は、その配送先のための対応するキュー１５を作成し、与えられたメッセージハンドラ４を登録する。
【００７９】(7)メッセージ５の不着検出のための属性を設定するメソッド
［定義］メッセージ５の不着検出のため、メッセージ５が送信される予定をメッセージハンドラ４に設定するためのメソッド。
【００８０】［シグネチャ］対象となるメッセージ５と、その送信に関する予定を表すオブジェクト。このオブジェクトには、そのメッセージ５に関する定期性や予定時刻などが設定されている。
【００８１】［動作］予定を表すオブジェクトの情報に従って、メッセージが未着の場合にメッセージ空間３の検索機能（後述）を用いて、所望のメッセージ５の情況を調べる。未着の場合は対応するメッセージハンドラ４の不着検出時に呼ばれるメソッド（後述）を呼び出す。
【００８２】メッセージ送受信のためのメソッド(B)：
(1)メッセージ５を送信するためのメソッド
［定義］メッセージ空間３に対して、メッセージ送信オブジェクト１がメッセージ５を送信するためのメソッド。
【００８３】［シグネチャ］送信するメッセージ５を引数としてもつ。
【００８４】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられたメッセージ５の種類とそれを特定する識別子、それが送信された時間をメッセージ空間データベース１０へ記録するとともに、与えられたメッセージ５を、それを担当するチャネル８に渡す。チャネル８は自分の種類に応じた方式でメッセージ配信を行う。
【００８５】(2)送達確認を送信するためのメソッド
［定義］メッセージ空間３に対して、メッセージ受信オブジェクト２が、受け取ったメッセージ５の送達確認を送信するためのメソッド。
【００８６】［シグネチャ］送信する送達確認を引数として持つ。
【００８７】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられた送達確認のもととなるメッセージ５を担当するチャネル８に渡す。チャネル８は自分の種類に応じた方式で送達確認を配信する。
【００８８】(3)送達確認をpull形式で受信するためのメソッド
［定義］メッセージ空間３に対して、メッセージ送信オブジェクト１がメッセージ５の送達確認をpull形式で受信するためのメソッド。
【００８９】［シグネチャ］送信したメッセージ５を引数としてもつ、オプションで、時間やメッセージ５を特定するための識別子を引数として渡す場合もある。返り値は送達確認。
【００９０】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は与えられたメッセージ５を扱うチャネル８に対して、与えられたメッセージ５に対応する送達確認を要求する。対応する送達確認がチャネル８に既に届けられていれば、それを返す。まだ届けられていなければ、空を返す。チャネル８の詳細な動作はその種類に依存する。時間やメッセージ５を特定するための識別子が渡された場合は、その値に応じたメッセージ５を配送する。
【００９１】(4)メッセージ５をpull形式で受信するためのメソッド
［定義］メッセージ空間３に対して、メッセージ受信オブジェクト２がメッセージ５をpull形式で受信するためのメソッド。
【００９２】［シグネチャ］必要に応じて、メッセージ受信オブジェクト２を特定するための識別子を引数としてもつ。オプションで、時間やメッセージ５を特定するための識別子を引数として渡す場合もある。返り値はメッセージ５。
【００９３】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は与えられたメッセージ５を扱うチャネル８に対して、メッセージ５の要求を伝える。チャネル８はその種類に応じた方式でメッセージ受信オブジェクト２にメッセージを配送する。時間やメッセージ５を特定するための識別子が渡された場合は、その値に応じたメッセージ５を配送する。
【００９４】メッセージ検索のためのメソッド(C)：
(1)メッセージ空間３でやり取りされているメッセージ５の種類の一覧を返すメソッド
［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）が、現在メッセージ空間３内でやり取りされているメッセージ５の種類の一覧を問い合わせるためのメソッド。
【００９５】［シグネチャ］問い合わせる際の条件式を引数としてもつ。この条件式は省略可能。返り値は、現在メッセージ空間３内でやり取りされている。条件式に該当するメッセージ５の一覧。条件式が省略されている場合は、全メッセージ５の一覧。
【００９６】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、現在登録されているメッセージ５のうち条件式に合致するものをメッセージ空間データベース３に問い合わせ、一覧として返す。条件式が省略されている場合は、現在やり取りされている全メッセー

ジ５の一覧を作成し、返す。
【００９７】(2)指定されたメッセージ５の送信オブジェクト一覧を返すメソッド
［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）が、現在メッセージ空間３内でやり取りされている任意のメッセージ５の送信オブジェクトの一覧を問い合わせるためのメソッド。
【００９８】［シグネチャ］送信オブジェクト一覧を調べたいメッセージ５を引数としてもつ。返り値は、現在のメッセージ空間３に指定されたメッセージ５を送信するメッセージ送信オブジェクト１の一覧。
【００９９】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられたメッセージ５のメッセージ送信オブジェクト１をメッセージ空間データベース１０に問い合わせ、一覧として返す。
【０１００】(3)指定されたメッセージ５の受信オブジェクト一覧を返すメソッド
［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）が、現在メッセージ空間３内でやり取りされている任意のメッセージ５の受信オブジェクトの一覧を問い合わせるためのメソッド。
【０１０１】［シグネチャ］受信オブジェクト一覧を調べたいメッセージ５を引数としてもつ。返り値は、現在のメッセージ空間３に指定されたメッセージ５を受信するメッセージ受信オブジェクト２の一覧。
【０１０２】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられたメッセージ５のメッセージ受信オブジェクト２をメッセージ空間データベース１０に問い合わせ、一覧として返す。
【０１０３】(4)指定されたメッセージ５の履歴一覧を返すメソッド
［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１およびメッセージ受信オブジェクト２）が、これまでにメッセージ空間３内でやり取りされていた任意のメッセージ５の一覧を問い合わせるためのメソッド。
【０１０４】［シグネチャ］履歴一覧を調べたいメッセージ５を引数としてもつ。返り値は、これまでにメッセージ空間３でやり取りされ、指定されたメッセージ５の一覧。
【０１０５】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられたメッセージ５の履歴情報をメッセージ空間データベース１０に問い合わせ、一覧として返す。
【０１０６】(5)指定されたメッセージ５が指定された時間以降にやり取りされているかどうかを調べるためのメソッド。
【０１０７】［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）が、あるメッセージ５について指定した時間以降にやり取りされているかどうかを調べるためのメソッド。
【０１０８】［シグネチャ］調べたいメッセージ５と指定する時間を引数として持つ。返り値は、やり取りされていたかどうかを示す２値。
【０１０９】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、与えられたメッセージ５が与えられた時間以降で、メッセージ空間３内でやり取りされた記録があるかどうかを、メッセージ空間データベース１０に問い合わせて調べ、その結果を返す。
【０１１０】利用終了のためのメソッド（D）：
(1)メッセージ空間３からの離脱を実施するメソッド
［定義］メッセージ空間３に対して、その利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）が、それ以降、メッセージ５をやり取りしないことを宣言するためのメソッド。
【０１１１】［シグネチャ］登録していたメッセージ５とその利用者（メッセージ送信オブジェクト１およびメッセージ受信オブジェクト２）を特定するための識別子を引数としてもつ。
【０１１２】［動作］このメソッドが呼ばれると、メッセージ空間管理部６は、メッセージ空間データベース１０における与えられた利用者の識別子に関する情報を削除するとともに、与えられたメッセージ５を扱っていたチャネル８に対して、与えられた識別子の利用者が、以後そのメッセージ５を扱わないこととするための処理を行う。キュー１５を用いていた場合は、そのキュー１５を始末する。
【０１１３】メッセージハンドラのためのメソッド（E）：
(1)メッセージ５が渡された際に呼び出されるメソッド
［定義］メッセージ５がメッセージ空間３へ送信された際に、push型でそのメッセージ５を受け取る場合に利用されるメッセージハンドラ４が実装するメソッド。
【０１１４】［シグネチャ］メッセージ５を引数にもつ。
【０１１５】［動作］このメソッドの動作はアプリケーション開発者に委ねられる。
【０１１６】(2)メッセージ５の不着が検出された際に呼び出されるメソッド
［定義］メッセージ５の不着が検出された際に呼び出される、メッセージハンドラ４が実装するメソッド。
【０１１７】［シグネチャ］エラーメッセージを含むオブジェクトを引数にもつ。
【０１１８】［動作］このメソッドの動作はアプリケーション開発者に委ねられる。
【０１１９】以下に、システム利用者（メッセージ送信オブジェクト１及びメッセージ受信オブジェクト２）か

らの利用シーケンスについて説明する。
【０１２０】図８は、システム利用者からみた本実施の形態におけるメッセージ空間３を利用するための流れ図である。この図８に示すように、初期化(ST801)、運用(ST802)、終了処理(ST803)の順に実行される。
【０１２１】以下に、メッセージ送信オブジェクト１からの利用シーケンスについて説明する。
【０１２２】図９は、メッセージ送信オブジェクト１からみた初期化(ST801)に関する流れ図である。この図９に示すように、コンテキストの取得(ST901)、登録(ST902)、送達確認用のメッセージハンドラの登録(ST903)の順に実行される。送達確認用のメッセージハンドラの登録(ST903)はオプションである。
【０１２３】コンテキストの取得(ST901)では上記メソッド(A)-(1)を用い、登録(ST902)では上記メソッド(A)-(2)、(A)-(3)を用い、送達確認用のメッセージハンドラの登録(ST903)では上記メソッド(A)-(4)を用いる。
【０１２４】図１０は、メッセージ送信オブジェクト１からみた運用(ST802)におけるメッセージ５のやり取りに関する流れ図である。この図１０に示すように、メッセージの送信(ST1001)、送達確認の受信(ST1002)の順に実行される。送達確認の受信(ST1002)はオプションである。
【０１２５】メッセージの送信(ST1001)では上記メソッド(B)-(1)を用い、送達確認の受信(ST1001)では、pull形式で利用する場合には上記メソッド(B)-(3)を用い、push形式で利用する場合には上記メソッド(A)-(4)で登録したメッセージハンドラで実装する上記メソッド(E)-(1)が用いられる。
【０１２６】メッセージ送信オブジェクト１からみた運用(ST802)におけるメッセージ検索については、上記で定義した処理(C)の単一の処理となるので、特に流れ図は定義しない。
【０１２７】メッセージ送信オブジェクト１からみた終了処理(ST803)では、上記で定義したメソッド(D)-(1)を用いる単一の処理となるので、特に流れ図は定義しない。
【０１２８】以下に、メッセージ受信オブジェクト２からの利用シーケンスについて説明する。
【０１２９】図１１は、メッセージ受信オブジェクト２からみた初期化(ST801)に関する流れ図である。この図１１に示すように、登録(ST1101)、メッセージハンドラの登録(ST1102)の順に実行される。メッセージハンドラの登録(ST1102)はオプションである。
【０１３０】登録(ST1101)では上記メソッド(A)-(5)を用い、メッセージハンドラの登録(ST1102)では上記メソッド(A)-(6)を用いる。
【０１３１】図１２は、メッセージ受信オブジェクト２からみた運用(ST802)におけるメッセージ５のやり取りに関する流れ図である。この図１２に示すように、受信(ST1201)、送達確認の送信(ST1202)の順に実行される。送達確認の送信(ST1202)はオプションである。
【０１３２】受信(ST1201)では、pull形式で利用する場合には上記メソッド(B)-(4)を用い、push形式で利用する場合には上記メソッド(A)-(6)で登録したメッセージハンドラで実装する上記メソッド(E)-(1)が用いられる。送達確認の送信(ST1202)では上記メソッド(B)-(2)を用いる。
【０１３３】メッセージ受信オブジェクト２からみた運用(ST802)におけるメッセージ検索については上記で定義した処理(C)の単一の処理となるので、特に流れ図は定義しない。
【０１３４】メッセージ受信オブジェクト２からみた終了処理(ST803)では、上記で定義したメソッド(D)-(1)を用いる単一の処理となるので、特に流れ図は定義しない。
【０１３５】以下に、本実施の形態に係るメッセージングシステムの仕組みを実現するための参照モデルについて説明する。以下で説明する分散システムのための目メッセージングシステムの参照モデルは、上述した構成やインタフェース定義を前提としている。
【０１３６】図１３は、メッセージングシステムのクラス図である。表記にはUMLを用いている。このクラス図を構成するクラスについてそれぞれ説明する。
【０１３７】メッセージ送信部２０は、本実施の形態に係るメッセージングシステムにおける参照モデルを利用し、メッセージ５を送信するアプリケーションである。このメッセージ送信部２０は、上記図１のメッセージ送信オブジェクト１に相当する。
【０１３８】メッセージ受信部２１は、本実施の形態に係るメッセージングシステムにおける参照モデルを利用し、メッセージ５を受信するアプリケーションである。このメッセージ受信部２１は、上記図１のメッセージ送信オブジェクト２に相当する。
【０１３９】メッセージ検索インタフェース２２は、メッセージ空間３におけるメッセージ検索を実現するためのインタフェース定義であり、上記メソッド(C)を定義する。
【０１４０】送信側インタフェース２３は、メッセージ送信部２０が利用するメッセージ空間３に対するインタフェース定義である。
【０１４１】受信側インタフェース２４は、メッセージ受信部２１が利用するメッセージ空間３に対するインタフェース定義である。
【０１４２】送信側管理部２５は、メッセージ送信部２０がメッセージ空間３とやり取りするための送信側の様々な処理を行うクラスであり、送達確認の不着調査も担当する。
【０１４３】受信側管理部２６は、メッセージ受信部２１がメッセージ空間３とやり取りするための受信側の様

々な処理を行うクラスであり、メッセージの不着調査も担当する。

【０１４４】メッセージ空間管理部２７は、メッセージ空間３の中心となるクラスである。このメッセージ空間管理部２７は、送信側管理部２５や受信側管理部２６とのやり取りを行い、初期化処理やメッセージ送受信の際のデータベースへの記録保持や適切なチャネルの振り分け、そして終了処理などを行う。このメッセージ空間管理部２７は、上記図１のメッセージ空間管理部６に相当する。

【０１４５】メッセージ空間データベース２８は、メッセージ空間３が保持する各種データを保存するためのクラスである。保持するデータには、例えばチャネル３０（このチャネルが利用するメッセージプール３１やチャネルマネージャ２９なども含む）の種類とメッセージ送信部２０及びメッセージ受信部２１の対応関係を示すデータ、メッセージ５毎の履歴情報などがある。メッセージ空間データベース２８は、上記図1のメッセージ空間データベース１０に相当する。

【０１４６】チャネルマネージャ２９は、少なくとも一つのチャネル３０を管理するためのクラスである。このチャネルマネージャ２９は、上記図1のチャネルマネージャ７に相当する。

【０１４７】チャネル３０は、メッセージの種類毎に用意されてメッセージのやり取りを管理するクラスである。チャネル３０は、送達確認処理を実現するためのメッセージと送達確認の対応関係に関する情報を保持する。このチャネル３０は、上記図1のチャネル８に相当する。

【０１４８】メッセージプール３１は、メッセージ５をメッセージ空間内に蓄えておくためのクラスである。このメッセージプール３１は、上記図１のメッセージプール９に相当する。

【０１４９】メッセージ３２は、本実施の形態においてやり取りされるメッセージ５を示すクラスである。メッセージ３２は、上記図1のメッセージ５に相当する。

【０１５０】送達確認３３は、メッセージ５の送達確認に利用される情報を表すクラスであり、メッセージ３２を継承する。

【０１５１】コンテキスト３４は、あるメッセージ５のやり取りが、メッセージ空間３内でどのように実現されているかを表すクラスである。

【０１５２】送達確認用メッセージハンドラ３５は、送達確認３３をpush型で受け取るためのクラスである。この送達確認用メッセージハンドラ３５は、上記図１のメッセージハンドラ４に相当する。

【０１５３】メッセージハンドラ３６は、メッセージ３２をpush型で受け取るためのクラスである。このメッセージハンドラ３６は、上記図１のメッセージハンドラ４に相当する。

【０１５４】これらのクラスのうち、チャネルマネージャ２９、チャネル３０、メッセージプール３１は、上記図２、３、７に示したように派生して利用されることが考えられる。この派生の例を図１４～図１６に示す。

【０１５５】また、代表的なシナリオに対するシーケンス図を示す。ただし、チャネル３０から先の動作はチャネル３０の種類／性質によって異なるため、それ以降の詳細なシーケンスはここでは省略する。

【０１５６】図１７は、メッセージ送信部２０による初期化のシーケンス図である。図１８は、メッセージ受信部２１による初期化のシーケンス図である。図１９は、メッセージ送信部２０によるメッセージ送信のシーケンス図である。図２０は、メッセージ受信部２１によるpush型のメッセージ受信のシーケンス図である。図２１は、メッセージ受信部２１によるpull型のメッセージ受信のシーケンス図である。図２２は、メッセージ送信部２０による終了処理のシーケンス図である。なお、メッセージ受信部２１による終了処理のシーケンス図も上記図２２と同様である。

【０１５７】以下に、分散システムのメッセージ交換でエラーが発生した場合のエラーハンドリングについて説明する。

【０１５８】図２３は、上記のようなメッセージングシステムにおいて考えられるエラーの発生箇所、種類を示すブロック図である。

【０１５９】メッセージ送信オブジェクト１における送信エラー、シリアライズの失敗などのようなエラーＥ１には、送信側管理部２５が対処の責任を負うとする。

【０１６０】メッセージ空間３における復元の失敗、受信エラーなどのようなエラーＥ２、送信エラー、シリアライズの失敗などのようなエラーＥ３、メッセージプールのあふれなどのようなエラーＥ５には、メッセージ空間管理部２７が対処の責任を負うとする。

【０１６１】メッセージ受信オブジェクト２における復元の失敗、受信エラーなどのようなエラーＥ４には、受信側管理部２６が対処の責任を負うとする。

【０１６２】なお、具体的な対処方法はエラー毎に定める。

【０１６３】図２４は、本実施の形態に係るメッセージングを模式的に表したブロック図である。

【０１６４】この図２４では、メッセージの伝送及び送達確認メッセージの伝送にキューを用い、どちらもpush型の配送をする構成を例に表している。

【０１６５】メッセージ送信オブジェクト４６は、メッセージ空間インタフェース４７を通してメッセージ空間管理部６にメッセージを送信する。

【０１６６】メッセージ空間管理部６は、必要な情報を記録すると同時に、適切なチャネルマネージャ７に依頼してキューチャネル１２を生成し（既にこのメッセージに対応するキューチャネル１２が存在している場合、こ

のプロセスは省略される）、キューチャネル１２にメッセージを渡す。

【０１６７】キューチャネル１２は、配信先として登録してあるメッセージ受信オブジェクト４８の分だけメッセージを複製する。そして、キューチャネル１２は、各メッセージ受信オブジェクト４８用のキュー４９にメッセージを渡し、これと同時に図２５に示す形式で各宛先へメッセージを配信したことを記録する。

【０１６８】キュー４９は、受け取ったメッセージを自分に登録されているメッセージハンドラ５０を通してメッセージ受信オブジェクト４８に送る。

【０１６９】メッセージ受信オブジェクト４８は、メッセージを受け取ったことが確認できると、送達確認メッセージをメッセージ空間インタフェース４７経由でメッセージ空間管理部６へ送る。

【０１７０】メッセージ空間管理部６は、送達確認メッセージの送信原因となったメッセージを扱ったキューチャネル１２に受け取った送達確認メッセージを渡す。

【０１７１】キューチャネル１２では、送達確認メッセージを受け取ったことを図２５に示す形式で記録し、送達確認メッセージの送信原因となったメッセージ送信オブジェクト４６に対応する送達確認用のキュー５１に、送達確認メッセージを渡す。これを受け取った送達確認用キュー５１は登録されている送達確認用メッセージハンドラ５２を通してメッセージ送信オブジェクト４６に送達確認がなされたことを伝える。

【０１７２】図２６は、上記図２４に示すメッセージングシステムを上記図１３のクラス図に基づいて配置図を兼ねたオブジェクト図として表した図である。

【０１７３】部分５３が本実施の形態におけるメッセージングの仕組みに相当する。

【０１７４】この図２６では送信側５４、受信側５５、メッセージ空間３がそれぞれ分けて記載されているが、メッセージ空間３の実装の実体がどこにあるかは特に限定されない。すなわち、送信側５４や受信側５５とは異なるハードウェア上で実装される場合も考えられるし、送信側５４や受信側５５の一方もしくは双方と同じハードウェア上で実装されてもよい。図２６中のアプリ側通信実装部５６、５７とメッセージ空間通信実装部５８、５９は実装の際に利用する分散通信技術に依存する部分である。すなわち、アプリ側通信実装部５６、５７とメッセージ空間通信実装部５８、５９は、利用する分散通信技術に応じて生成される。

【０１７５】また、メッセージ空間３を管理するのは基本的にメッセージ空間管理部２７であるが、負荷分散やフォールトトレランスの観点から、メッセージ空間管理部２７の請け負う仕事を分散させることも考えられる。この様子を図２７に示す。ここではメッセージ空間管理部２７の仕事を分散させるメッセージ空間サブ管理部６０を備えている。サーバの多重化技術には、既存のクラスタリングなどの技術を用いる。

【０１７６】以上説明した本実施の形態に係る分散システムのためのメッセージングシステムにおいては、インタフェース定義とそれへのアクセス方法を定義した参照モデルを採用している。

【０１７７】これにより、この定義されたインタフェースとそれへのアクセス方法を利用することで、実装基盤などに依存しない統一的なメッセージングが可能になる。

【０１７８】また、統一的なメッセージングを可能とする参照モデルを定義したことで、実装技術に関係なく利用可能なモデルを明確にすることができる。このため、ソフトウェアの共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲を明確化できる。また、メッセージを交換するための仕組みや機能、実装方法を統一化することにより、ある基盤技術の実装で培った経験を他の基盤技術で容易に活用できる。

【０１７９】また、本実施の形態に係るメッセージングシステムにおいては、既存の実装技術では提供されないメッセージの送達確認機能やメッセージ空間に対する検索機能、メッセージ不着検出およびその対処機能を提供できる。

【０１８０】また、本実施の形態に係るメッセージングシステムは、階層型の構造をしている。これにより、例えば業務の細部の仕様変更、新しい業務の追加、ハードウェア構成の仕様変更などのような様々な規模の仕様変更に容易に対処できる。

【０１８１】特に、例えばCORBA、JavaRMI，Java及びECJなどのような既存の分散システム間通信の実装技術に本実施の形態に係るメッセージングシステムを採用することにより、内部構造に互換性を持たせることができる。これにより、既存の分散システム間通信の実装技術に関わるソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲の明確化できる。

【０１８２】なお、本実施の形態で説明したメッセージングシステムの機能は、コンピュータに実行させることのできるプログラムとし、例えば磁気ディスク（フロッピー（登録商標）ディスク、ハードディスク等）、光ディスク（ＣＤ－ＲＯＭ、ＤＶＤ等）、半導体メモリなどの記録媒体に書き込んでコンピュータに適用可能である。また、このプログラムを通信媒体により伝送してコンピュータに適用することも可能である。上記機能を実現するコンピュータは、このプログラムを読み込み、プログラムによって動作が制御されることにより、上述した処理を実行する。

【０１８３】（第２の実施の形態）本実施の形態においては、CORBAのための参照モデルのマッピング手法について説明する。

【０１８４】図２８は、分散通信技術にCORBAを使用した場合のオブジェクト図である。この図２８は、上記図

２６の分散通信技術にCORBAを用いた場合のpush型のオブジェクト図である。

【０１８５】また、図２９はpull型のオブジェクト図である。なお、分散通信技術によらない部分（図２６の構成要素２０、２３、６１、２５、２１、３６、２４、２６、２７、６２、６３）は省略している。

【０１８６】通信側５４と受信側５５とのイベントの配送にはCORBAのイベントサービスを用いる。通常、CORBAのイベントサービスではイベントの送信側オブジェクトをサプライヤと受信側オブジェクトをコンシューマと呼び、イベントチャンネルにサプライヤとコンシューマを登録することによってイベントチャンネルが通信を仲介する。

【０１８７】Push型のイベント通信の場合、イベント送信者であるPushSupplier６６はCORBAイベントサービスのEventChanel６７、ProxyPushConsumer６８、SupplierAdmin６９と通信し、PushSupplier６６がイベント送信者であることを登録する。

【０１８８】同様に、イベント受信者であるPushConsumer７０はEventChanel６７、ProxyPushSupplier７１、ConsumerAdmin７２と通信し、PushConsumer７０がイベント受信者であることを登録する。以後、PushSupplier６６がProxyPushConsumer ６８にイベントを登録（push）するとPushConsumer７０はProxyPushSupplier７１からイベントを受け取ることができる。

【０１８９】さらに、イベント配送にCORBAのイベントサービスを使用するのみならず、送信側５４とメッセージ空間３、受信側５５とメッセージ空間３の通信を実現する必要がある。送信側５４のアプリ側通信実装部５６とメッセージ空間通信実装部５８は、アプリ側通信実装部スケルトン７６、アプリ側通信実装部スタブ７７、メッセージ空間通信実装部スタブ７８及びメッセージ空間通信実装部スケルトン７９によってCORBAによる通信を行う。

【０１９０】同様に、受信側５５のアプリ側通信実装部５７とメッセージ空間通信実装部５９は、アプリ側通信実装部スケルトン８２、アプリ側通信実装部スタブ８３、メッセージ空間通信実装部スタブ８４およびメッセージ空間通信実装部スケルトン８５によってCORBAによる通信を行う。

【０１９１】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なCORBAのための参照モデルのマッピング手法を示した。これにより、CORBAを利用したシステムの開発が容易になる。また、上記第１の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばJavaRMIやJava及びECJなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲を明確化できる。

【０１９２】（第３の実施の形態）本実施の形態においては、JavaRMIのための参照モデルのマッピング手法について説明する。

【０１９３】図３０は、分散通信技術にJavaRMI を使用した場合の模式図である。この図３０は、上記図２６中における分散通信技術に依存する部分をJavaRMIで実現している。分散通信技術に依存しない部分についての具体的な説明を省略する。

【０１９４】分散通信を行うために、アプリケーション開発者は以下の操作を行う。

【０１９５】まず、上記図２６のアプリ側通信実装部５６、５７とメッセージ空間通信実装部５８、５９の各々について、rmicコンパイラを用いてアプリ側通信実装部_Skel９０、９１、アプリ側通信実装部_Stub９２、９３、メッセージ空間通信実装部_Skel９４、９５、メッセージ空間通信実装部_Stub９６、９７を生成する。

【０１９６】次に、アプリ側通信実装部５６、５７とメッセージ空間通信実装部５８、５９の各々を、メッセージ空間に依存するネームサーバであるrmiregistryサーバ９８に登録する。ここで、メッセージ空間通信実装部_Stub９６は、送信側のJavaVMで利用されるメッセージ空間通信実装部５８の分身である。このオブジェクトにより、アプリ側通信実装部５６がメッセージ空間３のメッセージ空間通信実装部５８のメソッドを呼び出す際に、ネットワークの存在を意識せずに直接メソッドを呼び出す感覚で実行できる。

【０１９７】また、アプリ側通信実装部５６は、rmiregistryサーバ９８に問い合わせることにより、通信相手であるメッセージ空間通信実装部５８を探すことが可能である。

【０１９８】一方、メッセージ空間通信実装部_Skel９４は、メッセージ空間のJavaVMに存在し、送信側のメッセージ空間通信実装部_Stub９６からの通信を受け付ける。

【０１９９】このように、JavaRMIでは、StubとSkelがそれぞれネットワーク接続の出入口として機能することで、アプリ側通信実装部５６とメッセージ空間通信実装部５８の分散通信を実現する。アプリ側通信実装部_Skel９０、９１とアプリ側通信実装部_Stub９２、９３とメッセージ空間通信実装部_Skel９５とメッセージ空間通信実装部_Stub９７についても同様である。

【０２００】次に、メッセージ空間インタフェース４７（２３、２４）が持つインタフェース定義をどのように実現するかについて具体的に発明する。ただし、「メッセージハンドラのためのメソッド(E)」については、分散通信とは直接関係しないので、説明を省略する。なお、分散通信においてメソッドの引数として渡されるメッセージハンドラ３６、送達確認用メッセージハンドラ３５、メッセージ３２及びコンテキスト３４は、インタ

フェースjava.io.Serializableを実装する必要がある。
【０２０１】メッセージ空間インタフェース４７は、メソッド定義の集合であり、その実装は定義しない。この中身は、送信側管理部２５と受信側管理部２６でそれぞれ実装される。
【０２０２】送信側管理部２５は、メッセージ空間インタフェース２３が受けたメッセージを、アプリ側通信実装部５６に素通しするだけなので、説明は省略する。また、受信側管理部２６も、メッセージ空間インタフェース２４が受けたメッセージをアプリ側通信実装部５７に素通しするだけなので、説明は省略する。
【０２０３】次に、アプリ側通信実装部５６、５７、メッセージ空間通信実装部５８、５９について説明する。
【０２０４】アプリ側通信実装部クラスは、メッセージ空間通信実装部オブジェクトを属性として持つ。コンストラクタが起動された時には、rmiregistory ９８に登録されているメッセージ空間通信実装部５８を探し、属性にセットする。見つけられなかった場合はエラー処理を行う。なお、メッセージ空間通信実装部５８、５９は、予めrmiregistory９８に登録されているものとする。
【０２０５】メッセージ空間通信実装部クラスは必ずjava.rmi.server.Unicast Remote Objectを継承し、またメッセージ空間管理部クラスを属性として持つ。
【０２０６】コンストラクタはjava.rmi.Remote Exceptionの例外を必ずスローし、起動された時にはスーパクラスのコンストラクタを起動して、引数として渡されたメッセージ空間管理部オブジェクトを属性にセットする。
【０２０７】アプリ側通信実装部５６、５７のメソッドでは、実行の際に例外が発生しうるので、必ずtry-catch文を用いて例外をキャッチする。また、メッセージ空間通信実装部５８、５９のメソッドでは、java.rmi.Remote Exceptionの例外を必ずスローするように定義する。
【０２０８】上記第１の実施の形態で述べた個々のインタフェースについて、アプリ側通信実装部５６、５７と、メッセージ空間通信実装部５８、５９がどのように動作するかを、以下に説明する。
【０２０９】利用開始のためのメソッド(A)：
(1)送信するメッセージ５のメッセージ空間３におけるコンテキストの取得
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２１０】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２１１】(2)送信するメッセージ５の種類を登録するメソッド（その１）
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２１２】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２１３】(3)送信するメッセージ５の種類を登録するメソッド（その２）
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２１４】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２１５】(4)送達確認をpush形式で受け取るための確認用ハンドラを登録するメソッド
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２１６】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２１７】(5)受信するメッセージ５の種類を登録するメソッド
まずアプリ側通信実装部５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５９側のメソッドが呼び出される。
【０２１８】次にメッセージ空間通信実装部５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５７に返す。エラーが起きた場合は、nullを返す。
【０２１９】(6)メッセージ５をpush形式で受け取るためのメッセージハンドラを登録するメソッド
まずアプリ側通信実装部５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５９側のメソッドが呼び出される。
【０２２０】次にメッセージ空間通信実装部５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５７に返す。エラーが起きた場合は、nullを返す。
【０２２１】(7)メッセージ５の不着検出のための、属性を設定するメソッド
アプリ側通信実装部５７において、所定のメソッドが実行される。この中で、スケジュールの設定を行う。メッセージ通信は行わない。
【０２２２】メッセージ送信のためのメソッド（B）：

(1)メッセージ５を送信するためのメソッド
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２２３】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２２４】(2)送達確認を送信するためのメソッド
まずアプリ側通信実装部５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５９側のメソッドが呼び出される。
【０２２５】次にメッセージ空間通信実装部５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５７に返す。エラーが起きた場合は、nullを返す。
【０２２６】(3)送達確認をpull形式で受信するためのメソッド
まずアプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８側のメソッドが呼び出される。
【０２２７】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６に返す。エラーが起きた場合は、nullを返す。
【０２２８】(4)メッセージ５をpull形式で受信するためのメソッド
まずアプリ側通信実装部５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５９側のメソッドが呼び出される。
【０２２９】次にメッセージ空間通信実装部５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５７に返す。エラーが起きた場合は、nullを返す。
【０２３０】メッセージ検索のためのメソッド(C)：
(1)メッセージ空間３でやり取りされているメッセージ５の種類の一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２３１】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２３２】(2)指定されたメッセージ５の送信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２３３】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２３４】(3)指定されたメッセージ５の受信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２３５】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２３６】(4)指定されたメッセージ５の履歴一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２３７】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２３８】(5)指定されたメッセージ５が指定された時間以降にやり取りされているかどうかを調べるためのメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２３９】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２４０】利用終了のためのメソッド(D)：
(1)メッセージ空間３からの離脱を実施するメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、メッセージ空間通信実装部５８、５９側のメソッドが呼び出される。
【０２４１】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果をアプリ側通信実装部５６、５７に返す。エラーが起きた場合は、nullを返す。
【０２４２】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なJavaRMIのための参照モデルのマッピング手法を示した。これにより、JavaRMIを利用したシステムの開発が容易になる。また、上記第１の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばCORBAやJava及びECJなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範

囲を明確化できる。

【０２４３】（第４の実施の形態）本実施の形態においては、ECJのための参照モデルのマッピング手法について説明する。

【０２４４】図３１は、分散通信技術にECJを使用した場合の模式図である。この図３１は、上記図２６中における分散通信技術に依存する部分をECJで実現している。分散通信技術に依存しない部分についての具体的な説明を省略する。

【０２４５】ECJの主な構成と動作を図３２に示す。ECJは、イベントID属性とイベントデータ属性を持つイベントオブジェクト９９、イベントIDに対応する処理を行うハンドラ１００、イベントIDとそれに対応するイベントのハンドラ１００の対応表１０１を持つLocalManager１０２から構成される。外部からイベント９９を受け付けたLocalManager１０２は、自分が持つ対応表１０１に照らしあわせて、該当するハンドラ１００を起動する。アプリケーション開発者は、イベント９９に応じたハンドラ１００を作成する。

【０２４６】ECJはイベント配送型であるため、分散通信を行うに際して、あるアクションに対する返り値を取得する方法がない。そこで、図３３に示すWakeup方式を用いることにより、RMIと同様の操作を実現する。なおWakeup方式では、ECJのLocalManagerクラスを拡張し、ロックインスタンスとイベントオブジェクトの管理を実施可能としたECLMS Local Managerクラス、及びECJのUser Eventクラスを拡張し、イベントID属性とイベントデータ属性に加えて返り値イベントID属性を持つクラスECJ Message Eventクラスを定義して用いる。

【０２４７】これにより、例えば、イベントID0001と、引数となるイベントデータと、返り値イベントID1000を持つイベントを発生させることにより、Wakeup Handlerによって、イベントID1000と、返り値となるイベントデータを持つイベント353を確実に発生させ、返り値を取得することが可能になる以下にWakeup方式の流れ（上記図３１、図３３）について説明する。例として、ID0001のイベントを発生させて、返り値としてID1000のイベントが発生するものとする。

(1)送信側５４のアプリ側通信実装部５６は、イベントID1000とWakeup Handler１０３を、Local Manager １０４に登録する。さらに、ID1000のロックインスタンスを生成し、Local Manager１０４に登録する。

【０２４８】(2)イベントIDが0001、イベントデータが引数となるデータ、返り値イベントIDが1000であるイベント１０５を、IPマルチキャストを通じてネットワーク１０６に配信する。

【０２４９】(3)送信側５４のアプリ側通信実装部５６は、IDが1000であるロックインスタンスが存在している間はsleepする。

【０２５０】(4)メッセージ空間３のLocal Manager１０７に、イベントID0001のイベント１０５が配送される。

【０２５１】(5)Local Manager１０７は、イベントID0001に対応するハンドラを起動する。起動されたハンドラは何らかの処理を実行したのち、イベントID1000のイベントを生成して、IPマルチキャストを通じてネットワーク１０６に配信する。

【０２５２】(6) 送信側５４のアプリ側通信実装部５６のLocal Manager１０４に、イベントID1000のイベントが配送される。

【０２５３】(7)Local Manager１０４は、イベントID1000に対応するWakeup Handler１０３を起動する。

【０２５４】この中でWakeup Handler１０３は、Local Manager１０４にイベントID1000のイベントを保存し、イベントIDと同じIDのロックインタンスを消去する。

【０２５５】(8) 送信側５４のアプリ側通信実装部５６は、上記(3)のsleepからぬけて、処理を再開する。Local Manager１０７が保持するイベントID1000のイベントのイベントデータをもとに処理を進める。

【０２５６】(9)イベントID1000のイベントを削除し、イベント登録を解除する。

【０２５７】分散通信を実施するために、アプリケーション開発者は以下の操作を行う。

【０２５８】まず、ECJのクラスを用いて、LocalManager１０４、１０７～１０９、WakeupHandler１０３、１１０～１１２、イベント１０５、１１３～１１５を生成する。その後、各LocalManagerに対して、イベントIDとアプリ側通信実装部５６、５７、メッセージ空間通信実装部５８、５９を登録する。ここでは、アプリ側通信実装部５６、５７、メッセージ空間通信実装部５８、５９がハンドラとしての役割を持つ。このように、ECJを用いた場合は、ECJ本来の機能に加えてWakeup方式を用いることで、アプリ側通信実装部とメッセージ空間通信実装部の分散通信を実現する。

【０２５９】次に、メッセージ空間インタフェース４７が持つ上記第１の実施の形態で述べたインタフェース定義をどのように実現するかについて具体的に説明する。ただし、「メッセージハンドラのためのメソッド(E)」については、分散通信とは直接関係しないので、説明を省略する。

【０２６０】主な通信は上記で述べたWakeup方式を用いる。また、ECJの機能で実現できる部分に関しては、特に明示的な通信を行わない場合もある。上記図２６におけるメッセージ空間インタフェース２３、２４、送信側管理部２５、受信側管理部３６は、RMIマッピングにおけるものと同じであるので説明を省略する。

【０２６１】次に、アプリ側通信実装部５６、５７、メッセージ空間通信実装部５８、５９について説明する。

【０２６２】アプリ側通信実装部クラスは、LocalManager１０４、１０８とトランスポートアダプタを属性として持つ。トランスポートアダプタは、IPマルチキャスト

を実現するうえで必要になる。コンストラクタが起動された時には、新しいLocalManager１０４、１０８が生成され、属性にセットされる。また、新しいターミナルアダプタが生成され、属性にセットされる。生成に失敗した場合はエラー処理を行なう。

【０２６３】メッセージ空間通信実装部５８、５９のクラス定義は、アプリ側通信実装部クラスと同じなので、説明を省略する。また、メッセージ空間通信実装部クラスはハンドラとして振る舞うので、executeメソッドをオーバライドする。executeメソッドの中で、switch-case文を用いることにより、イベントに対する適切なメソッドを起動することが可能である。なお、switch-case文に代えてメッセージ空間通信実装部クラスをサブクラス化して実現することも可能である。

【０２６４】アプリ側通信実装部５６、５７と、メッセージ空間通信実装部５８、５９の双方において、イベントを送信する時は、例外が発生しうるので、try-catch文を用いて、例外をキャッチする。

【０２６５】発生の構成の項で述べた個々のインタフェースについて、アプリ側通信実装部５６、５７と、メッセージ空間通信実装部５８、５９がどのように動作するかを以下に説明する。

【０２６６】利用開始のためのメソッド(A)：

(1)送信するメッセージ５のメッセージ空間３におけるコンテキストの取得

まずアプリ側通信実装部５６において、所定のメソッドがWekeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８側のメソッドが起動される。

【０２６７】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６においてWekeupHandlerが実行される。

【０２６８】その後、アプリ側通信実装部５６において実行された所定のメソッドは処理を再開し、結果を返す。エラーが起きた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【０２６９】(2)送信するメッセージ8の種類を登録するメソッド（その１）

まずアプリ側通信実装部５６において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８側のメソッドが起動される。次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６においてWakeup Handlorが実行される。

【０２７０】その後、アプリ側通信実装部５６において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【０２７１】(3)送信するメッセージ8の種類を登録するメソッド（その２）

まずアプリ側通信実装部５６において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８側のメソッドが起動される。

【０２７２】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６においてWakeupHandlerが実行される。その後、アプリ側通信実装部５６において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【０２７３】(4)送達確認をpush形式で受け取るための確認用ハンドラを登録するメソッド

アプリ側通信実装部５６において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager １０４にハンドラが登録される。

【０２７４】(5)受信するメッセージ５の種類を登録するメソッド

まずアプリ側通信実装部５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５９側のメソッドが起動される。

【０２７５】次にメッセージ空間通信実装部132において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５７においてWakeupHandlerが実行される。

【０２７６】その後、アプリ側通信実装部５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが起きた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【０２７７】(6)メッセージ５をpush形式で受け取るためのメッセージハンドラを登録するメソッド

アプリ側通信実装部５７において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager１０８にハンドラが登録される。

【０２７８】(7)メッセージ５の不着検出のための、属性を設定するメソッド

アプリ側通信実装部５７において、所定のメソッドが実行される。この中で、アプリ側のアプリケーションに対してスケジュールが登録される。

【０２７９】メッセージ送信のためのメソッド（B）：

(1)メッセージ５を送信するためのメソッド

アプリ側通信実装部５６において、所定のメソッドが実行される。この中で、メッセージをイベントとして配送

する。
【０２８０】(2)送達確認を送信するためのメソッド
アプリ側通信実装部５７おいて、所定のメソッドが実行される。この中で、メッセージをイベントとして配送する。
【０２８１】(3)送達確認をpull形式で受信するためのメソッド
まずアプリ側通信実装部５６において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８側のメソッドが起動される。
【０２８２】次にメッセージ空間通信実装部５８において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６においてWakeupHandlerが実行される。
【０２８３】その後、アプリ側通信実装部５６において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２８４】(4)メッセージ５をpull形式で受信するためのメソッド
まずアプリ側通信実装部５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５９側のメソッドが起動される。
【０２８５】次にメッセージ空間通信実装部５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５７において、Wakeup Handlerが実行される。
【０２８６】その後、アプリ側通信実装部５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２８７】メッセージ検索のためのメソッド(C)：
(1)メッセージ空間３でやり取りされているメッセージ５の種類の一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８、５９側のメソッドが起動される。
【０２８８】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６、５７においてWakeup Handlerが実行される。
【０２８９】その後、アプリ側通信実装部５６、５７において実行された所定のメソッドが処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２９０】(2)指定されたメッセージ8の送信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８、５９側のメソッドが起動される。
【０２９１】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６、５７においてWakeup Handlerが実行される。
【０２９２】その後、アプリ側通信実装部５６、５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２９３】(3)指定されたメッセージ8の受信オブジェクト一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８、５９側のメソッドが起動される。
【０２９４】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６、５７においてWakeup Handlerが実行される。
【０２９５】その後、アプリ側通信実装部５６、５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２９６】(4)指定されたメッセージ５の履歴一覧を返すメソッド
まずアプリ側通信実装部５６、５７において、所定のメソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８、５９側のメソッドが起動される。
【０２９７】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６、５７においてWakeup Handlerが実行される。
【０２９８】その後、アプリ側通信実装部５６、５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。
【０２９９】(5)指定されたメッセージ５が指定された時間以降のやり取りされているかどうかを調べるためのメソッド
まずアプリ側通信実装部５６、５７において、所定のメ

ソッドがWakeup方式で実行される。この中で送信されたイベントにより、メッセージ空間通信実装部５８、５９側のメソッドが起動される。

【０３００】次にメッセージ空間通信実装部５８、５９において、所定のメソッドが実行され、その実行結果がイベントとして送信され、アプリ側通信実装部５６、５７においてWakeup Handlerが実行される。

【０３０１】その後、アプリ側通信実装部５６、５７において実行された所定のメソッドは処理を再開し、結果を返す。エラーが生じた場合の処理や、通信におけるタイムアウトなどの処理は作成するアプリケーションに依存する。

【０３０２】利用終了のためのメソッド（D）：

(1)メッセージ空間3からの離脱を実施するメソッド

アプリ側通信実装部５６、５７において、所定のメソッドが実行される。この中で、アプリ側のLocal Manager１０４、１０８に対してメッセージハンドラを登録解除する。

【０３０３】以上のように、本実施の形態においては、既存の分散システム間通信の実装技術として代表的なECJのための参照モデルのマッピング手法を示した。これにより、ECJを利用したシステムの開発が容易になる。また、上記第１の実施の形態で説明したメッセージングシステムと同様のモデルを利用することで、例えばCORBAやJavaRMIなどのような他の分散システム間通信の実装技術との間で内部構造に互換性を持たせることもできる。したがって、ソフトウェア部品の共通化／再利用や保守／管理が容易になり、仕様変更の波及範囲を明確化できる。

【０３０４】

【発明の効果】以上詳記したように、本発明を利用してメッセージ交換の実装のための手段や仕組みを統一化することで、統一的なメッセージ交換のモデルを与えることができる。

【０３０５】これにより、実装技術が異なる場合であっても内部の構造に互換性を持たせることができる。したがって、ソフトウェアの共通化／再利用、保守／管理が容易となり、仕様を変更する場合に影響を受ける範囲を明確化できる。

【０３０６】また、本発明を利用して分散システムにおけるメッセージ交換のためのインタフェースやプロトコルを共通化することにより、実装基盤に依存しない統一的なメッセージングの仕組みを提供できる。

【０３０７】また、本発明ではメッセージ交換のための構造を階層型としているため、業務の細部の仕様変更、新しい業務の追加、ハードウェア構成の変更など、各種仕様変更に容易に対処できる。また、例えばメッセージの送達確認機能、メッセージ空間に対する検索機能、メッセージ不着信検出及びその対処機能を容易に実現させることができる。

【図面の簡単な説明】

【図１】本発明の第１の実施の形態に係るメッセージングシステムを利用する分散システムの構成を例示するブロック図。

【図２】チャネルマネージャの派生を示す図。

【図３】チャネルの派生を示す図。

【図４】キューチャネルによるメッセージ伝達の様子を示す図。

【図５】ストアチャネルによるメッセージ伝達の様子を示す図。

【図６】シンプルチャネルによるメッセージ伝達の様子を示す図。

【図７】メッセージプールの派生を示す図。

【図８】メッセージ空間を利用する場合の流れ図。

【図９】メッセージ送信オブジェクトからみた初期化の流れ図。

【図１０】メッセージ送信オブジェクトからみた運用の流れ図。

【図１１】メッセージ受信オブジェクトからみた初期化の流れ図。

【図１２】メッセージ受信オブジェクトからみた運用の流れ図。

【図１３】同実施の形態に係るメッセージングシステムの仕組みを実現するための参照モデルを例示するクラス図。

【図１４】チャネルマネージャクラスのサブクラス化を示す図。

【図１５】チャネルクラスのサブクラス化を示す図。

【図１６】メッセージプールのサブクラス化を示す図。

【図１７】メッセージ送信部による初期化のシーケンス図。

【図１８】メッセージ受信部による初期化のシーケンス図。

【図１９】メッセージ送信部によるメッセージ送信のシーケンス図。

【図２０】メッセージ受信部によるpush型のメッセージ受信のシーケンス図。

【図２１】メッセージ受信部によるpull型のメッセージ受信のシーケンス図。

【図２２】メッセージ送信部による終了処理のシーケンス図。

【図２３】エラーの発生箇所／種類とその対処を担当する部分の関係を示す図。

【図２４】同実施の形態に係るメッセージングシステムによるキューを用いたメッセージ交換の一例を示す図。

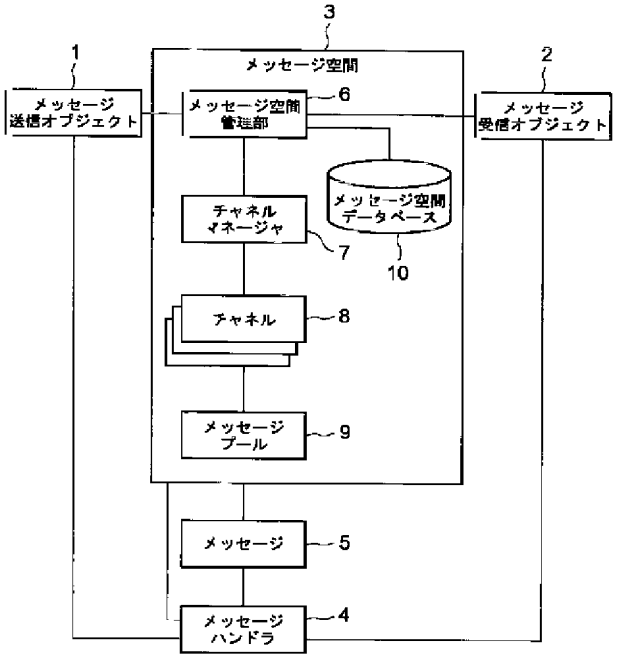
【図２５】キューチャネルで記録される情報を示した図である。

【図２６】同実施の形態に係るメッセージングシステムのオブジェクト図。

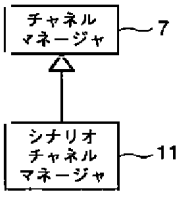
【図２７】本発明のメッセージ空間の機能を分散させた

場合のオブジェクト図である。
【図２８】CORBAを使用した場合のオブジェクト図（Push型）。
【図２９】CORBAを使用した場合のオブジェクト図（Pull型）。
【図３０】RMIマッピングを示す図。
【図３１】ECJマッピングを示す図。
【図３２】ECJ構成図。
【図３３】Wakeup方式のシーケンス図。
【符号の説明】
１…メッセージ送信オブジェクト
２…メッセージ受信オブジェクト
３…メッセージ空間
４…メッセージハンドラ
５…メッセージ
６…メッセージ空間管理部
７…チャネルマネージャ
８…チャネル
９…メッセージプール
１０…メッセージ空間データベース
４７…メッセージ空間インタフェース

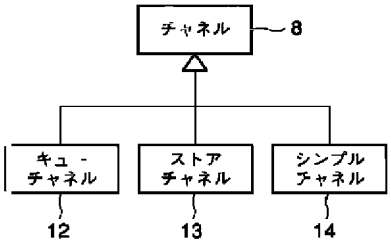
【図１】



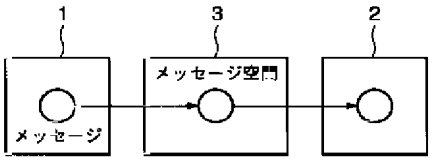
【図２】



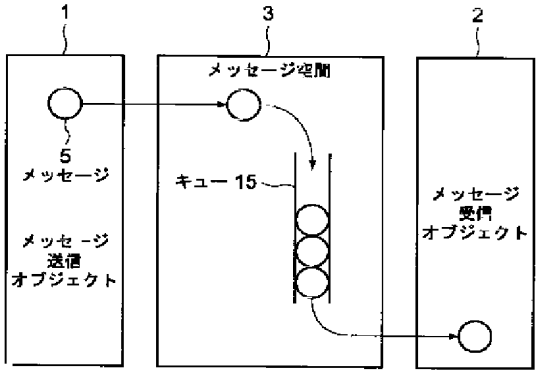
【図３】



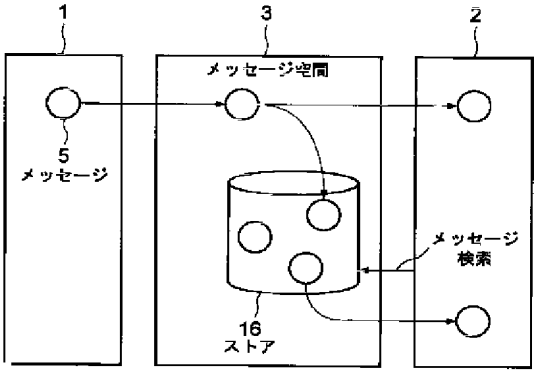
【図６】



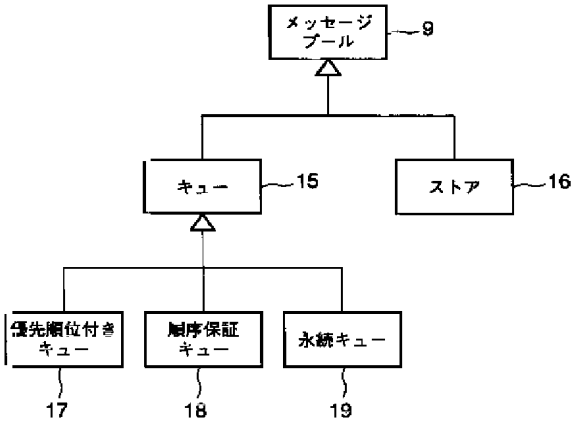
【図４】



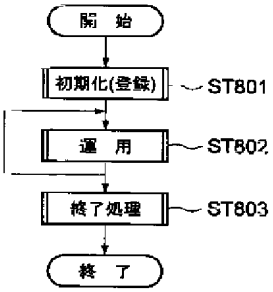
【図５】



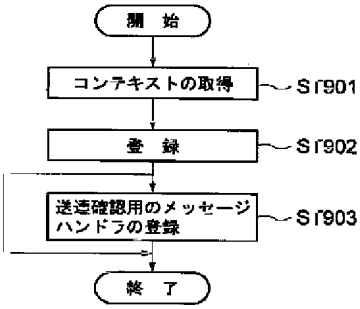
【図７】



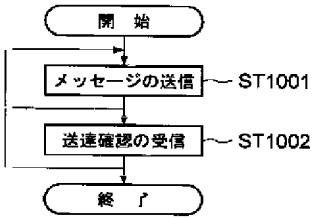
【図８】



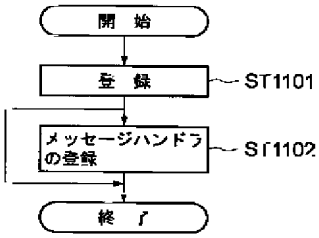
【図９】



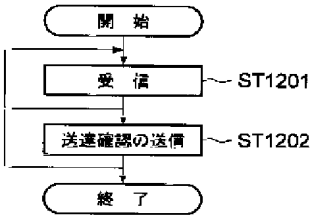
【図１０】



【図１１】

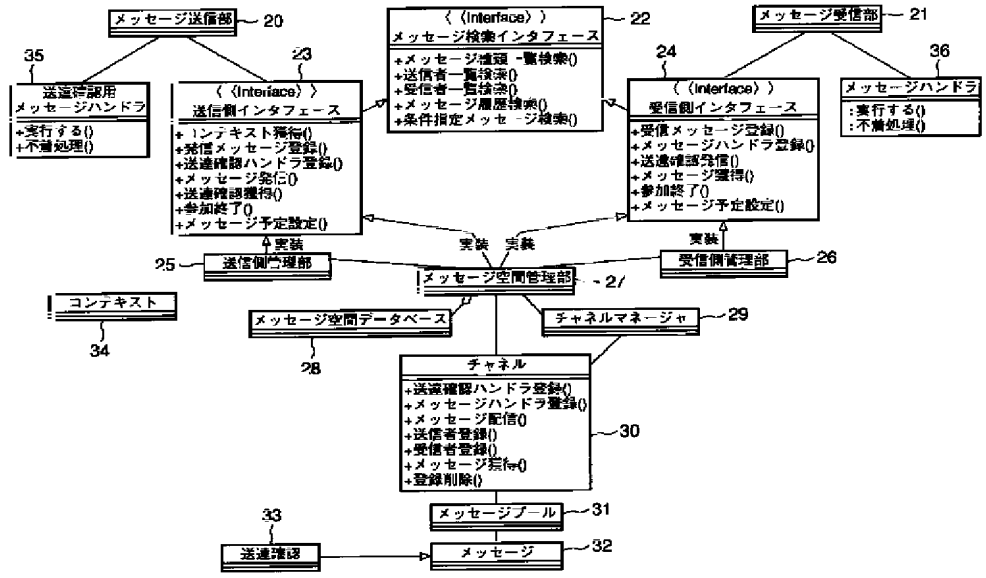


【図１２】

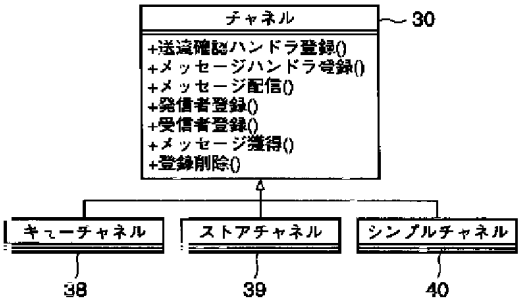


【図１４】

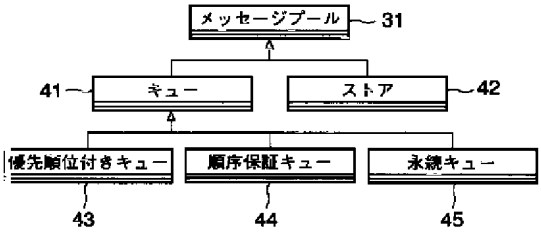
【図１３】



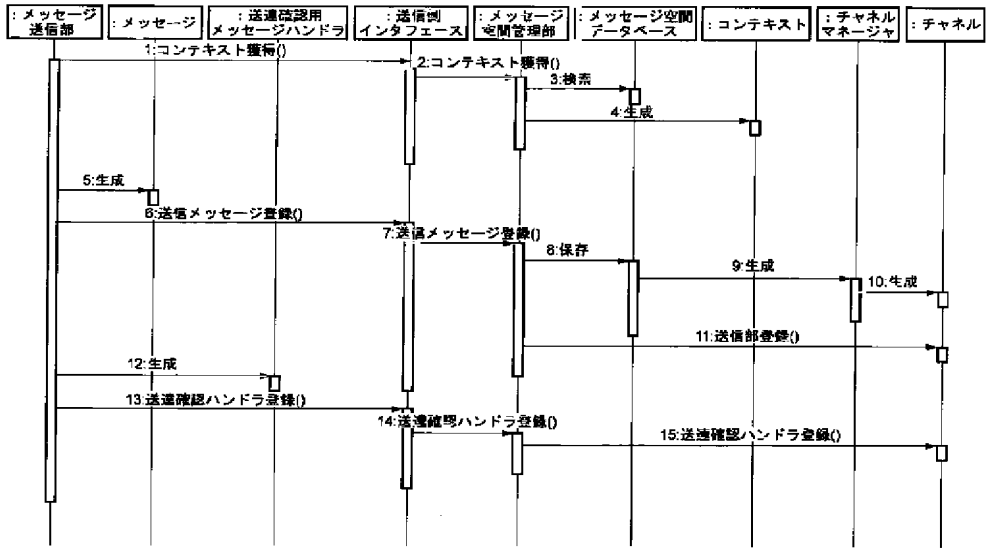
【図１５】



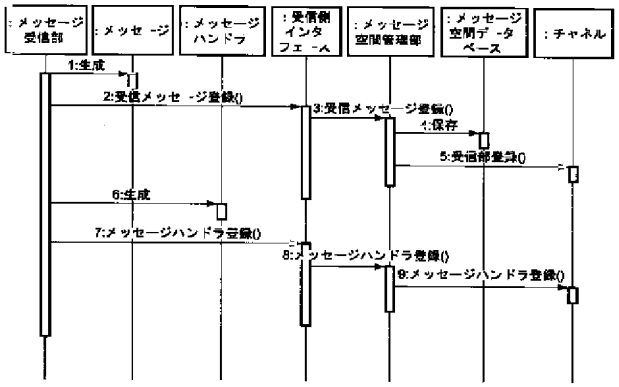
【図１６】



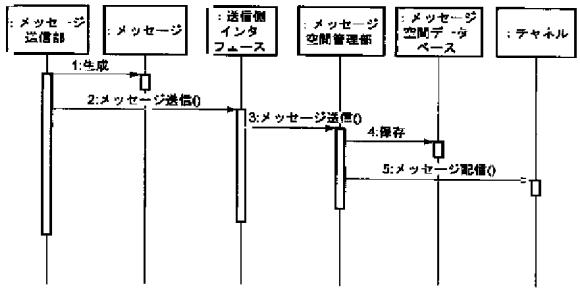
【図１７】



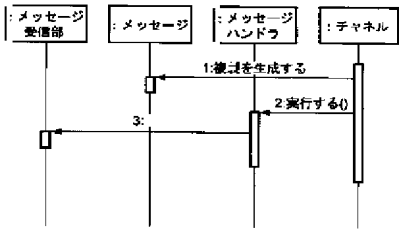
【図１８】



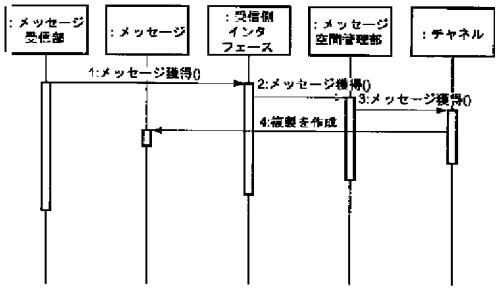
【図１９】



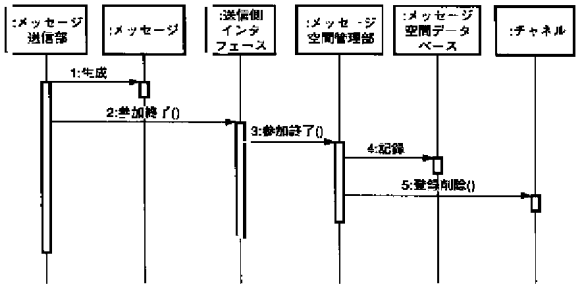
【図２０】



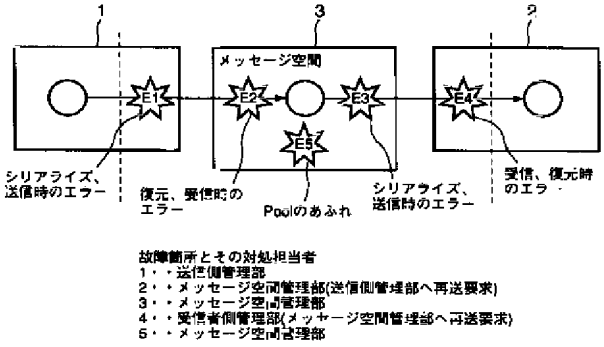
【図２１】



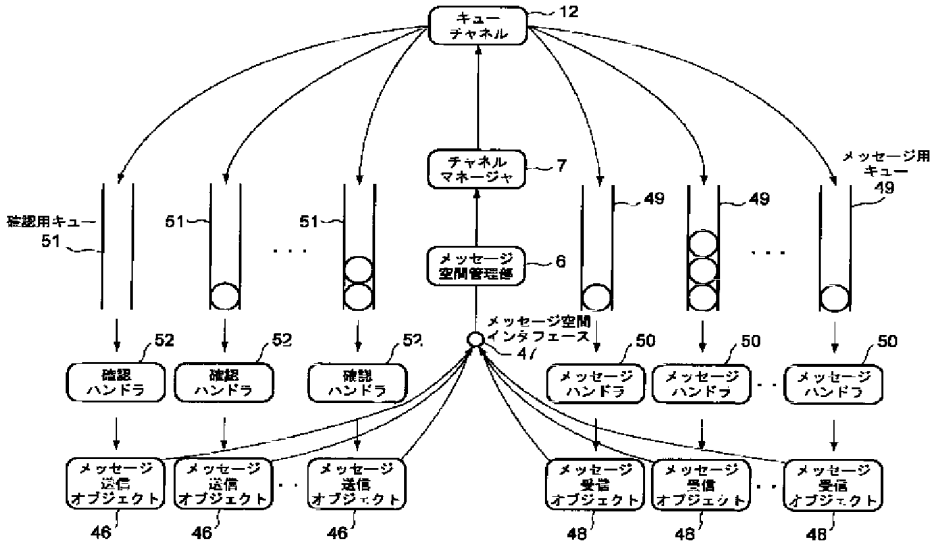
【図２２】



【図２３】



【図２４】

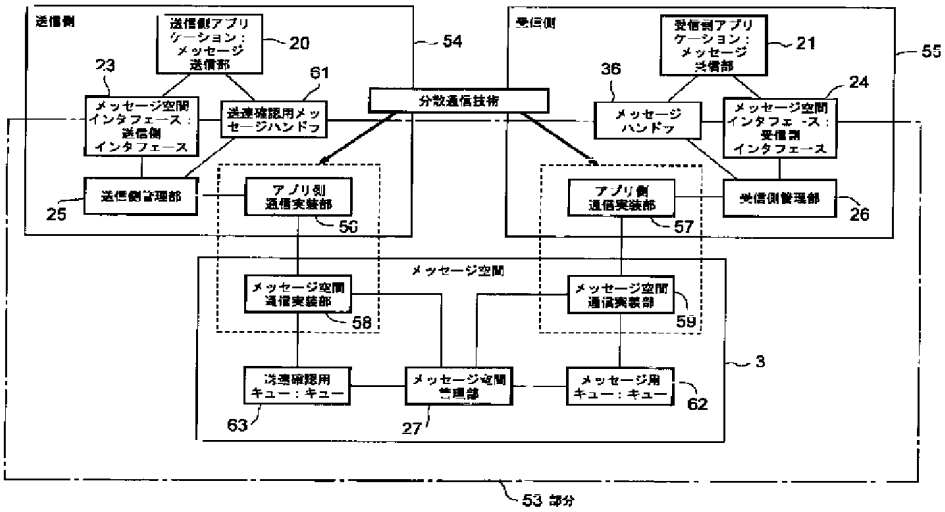


【図25】

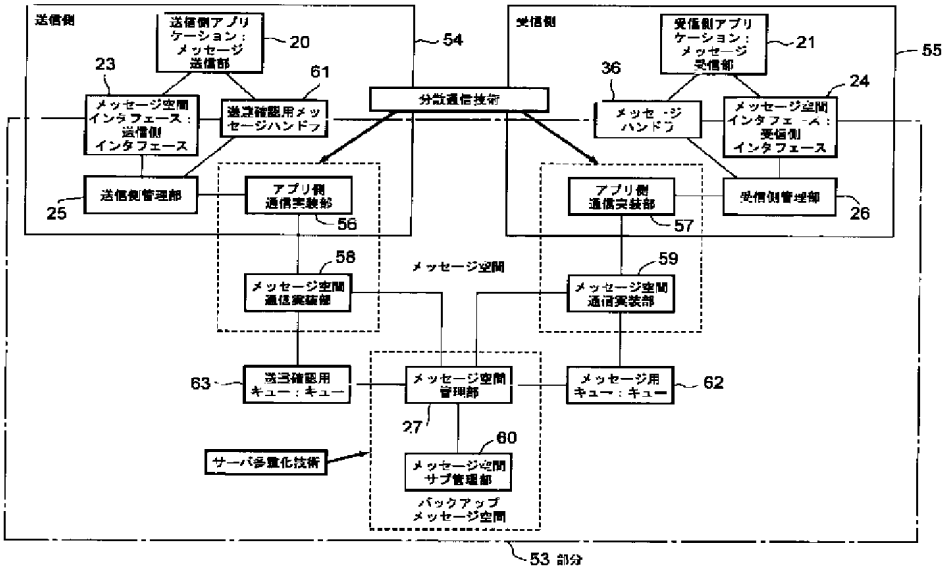
メッセージID	配信先1	配信先2	配信先3	・・・	情報
000001	確認済み	確認済み	確認済み	・・・	完了
000002	送信済み	確認済み	確認済み	・・・	未完
000003	送信済み	送信済み	送信済み	・・・	未完
：	：	：	：		：

キューチャネルで記録されるデータ

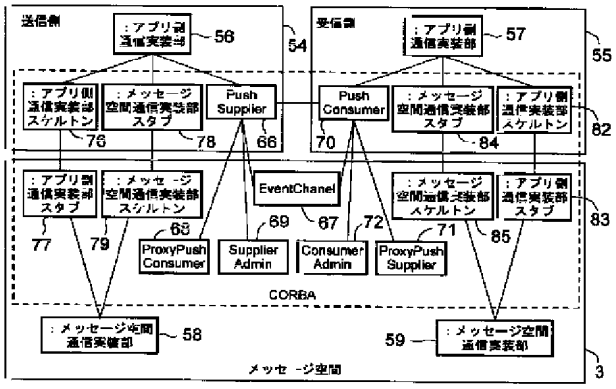
【図26】



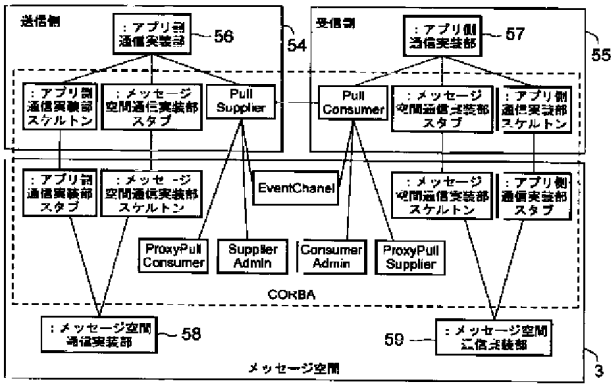
【図27】



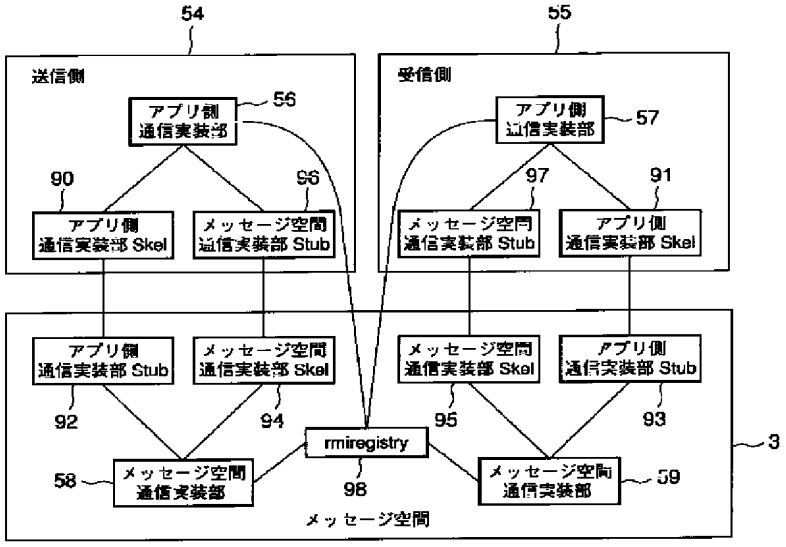
【図28】


送信側
受信側
Push Supplier
Push Consumer
EventChanel
ProxyPush Consumer
Supplier Admin
Consumer Admin
ProxyPush Supplier
CORBA
メッセージ空間

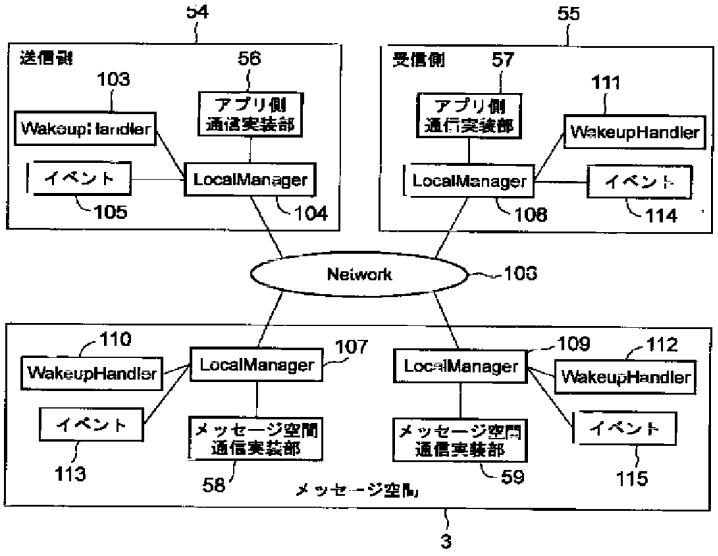
【図29】


送信側
受信側
Pull Supplier
Pull Consumer
EventChanel
ProxyPull Consumer
Supplier Admin
Consumer Admin
ProxyPull Supplier
CORBA
メッセージ空間

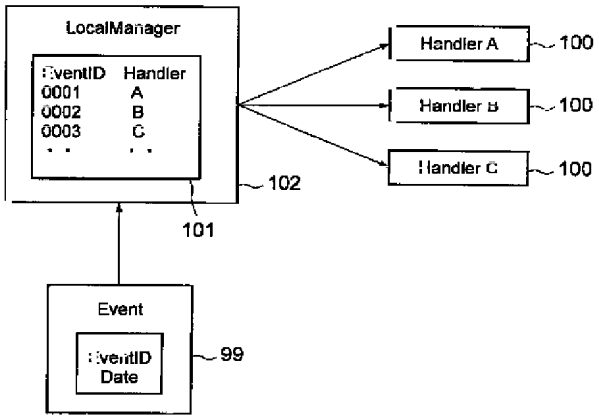
【図30】


54
55
送信側
受信側
アプリ側 通信実装部
アプリ側 通信実装部 Skel
メッセージ空間 通信実装部 Stub
メッセージ空間 通信実装部 Skel
アプリ側 通信実装部 Stub
rmiregistry
メッセージ空間
3

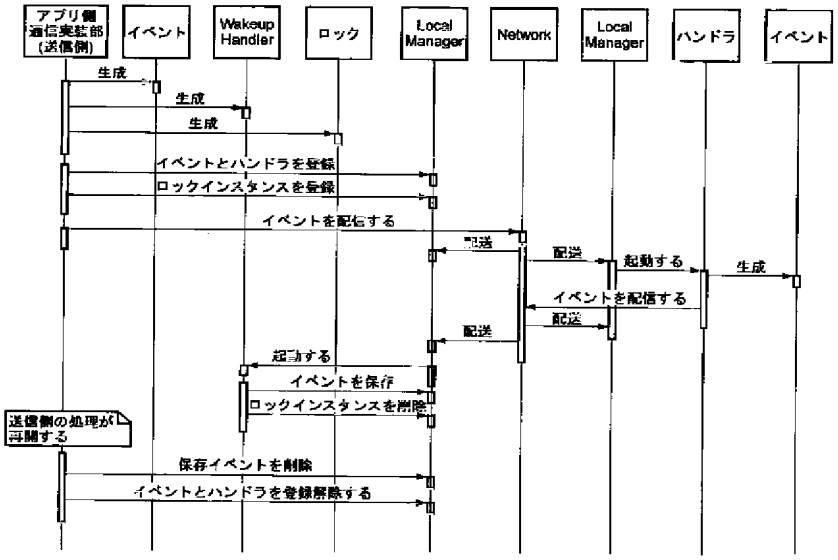
【図31】


54
55
送信側
56
103
アプリ側
通信実装部
WakeupHandler
イベント
LocalManager
105
104
受信側
57
111
アプリ側
通信実装部
WakeupHandler
LocalManager
イベント
108
114
Network
100
110
WakeupHandler
LocalManager
107
LocalManager
109
112
WakeupHandler
イベント
メッセージ空間
通信実装部
メッセージ空間
通信実装部
イベント
113
58
59
115
メッセージ空間
3

【図32】


LocalManager
EventID Handler
0001 A
0002 B
0003 C
Handler A
100
Handler B
100
Handler C
100
102
101
Event
EventID
Date
99

【図３３】



フロントページの続き

(72)発明者　秋元　直人
東京都府中市東芝町１番地　株式会社東芝府中事業所内

(72)発明者　鞍田　友美
東京都府中市東芝町１番地　株式会社東芝府中事業所内

Ｆターム(参考)　5B045 BB31 GG01
5B089 GB01 GB10 JA11 KA13 KB06
KB09 KE01